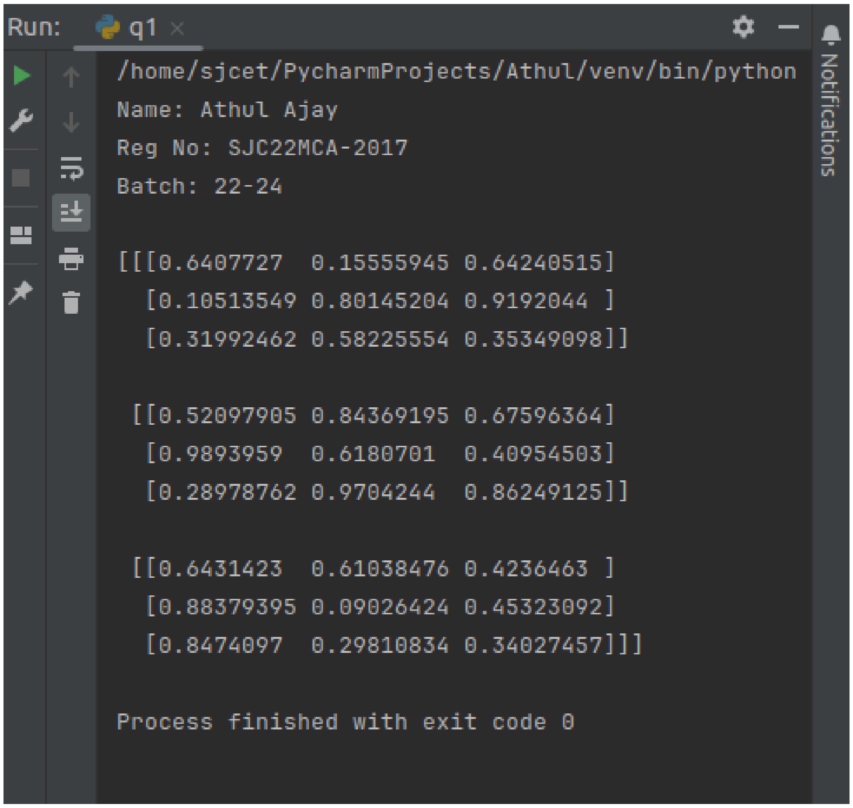
LAB CYCLE 2

1. Create a three-dimensional array specifying float data type and print it

```
import numpy as np

shape = (3, 3, 3)
d_type = np.float32
my_3d_array = np.random.rand(*shape).astype(d_type)
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
print(my_3d_array
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

[[[0.6407727  0.15555945 0.64240515]
  [0.10513549 0.80145204 0.9192044 ]
  [0.31992462 0.58225554 0.35349098]]

 [[0.52097905 0.84369195 0.67596364]
  [0.9893959  0.6180701  0.40954503]
  [0.28978762 0.9704244  0.86249125]]

 [[0.6431423  0.61038476 0.4236463 ]
  [0.88379395 0.09026424 0.45323092]
  [0.8474097  0.29810834 0.34027457]]]

Process finished with exit code 0
```

2. Create a 2-dimensional array (2X3) with elements belonging to complex data type and print it. Also display

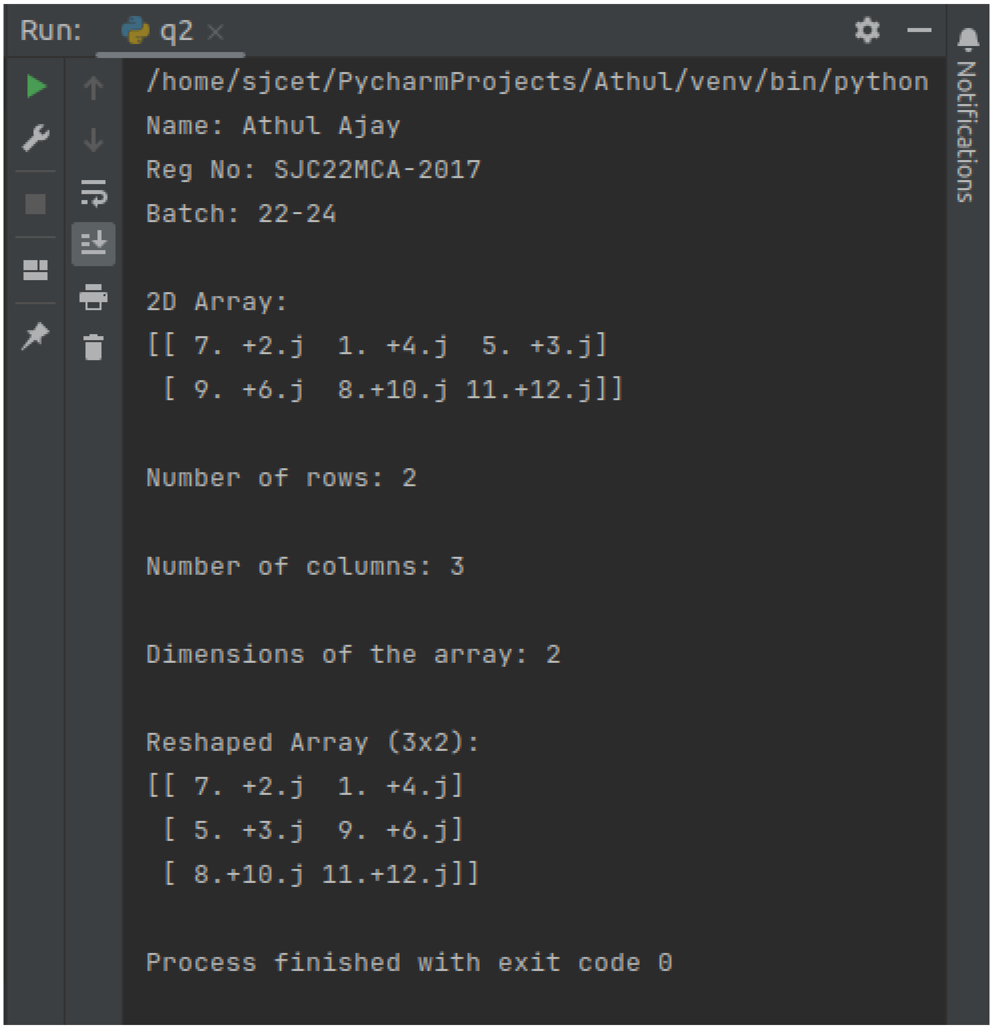
a. the no: of rows and columns

b. dimension of an array

c. reshapes the same array to 3X2

```
import numpy as np


complex_array = np.array([[7 + 2j, 1 + 4j, 5 + 3j],
                          [9 + 6j, 8 + 10j, 11 + 12j]], dtype=complex)

print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
print("2D Array:")
print(complex_array)


num_rows, num_columns = complex_array.shape


dimensions = complex_array.ndim


print("\nNumber of rows:", num_rows)
print("\nNumber of columns:", num_columns)
print("\nDimensions of the array:", dimensions)


reshaped_array = complex_array.reshape(3, 2)


print("\nReshaped Array (3x2):")
print(reshaped_array)
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

2D Array:
[[ 7. +2.j  1. +4.j  5. +3.j]
 [ 9. +6.j  8.+10.j 11.+12.j]]

Number of rows: 2

Number of columns: 3

Dimensions of the array: 2

Reshaped Array (3x2):
[[ 7. +2.j  1. +4.j]
 [ 5. +3.j  9. +6.j]
 [ 8.+10.j 11.+12.j]]

Process finished with exit code 0
```

3. Familiarize with the functions to create

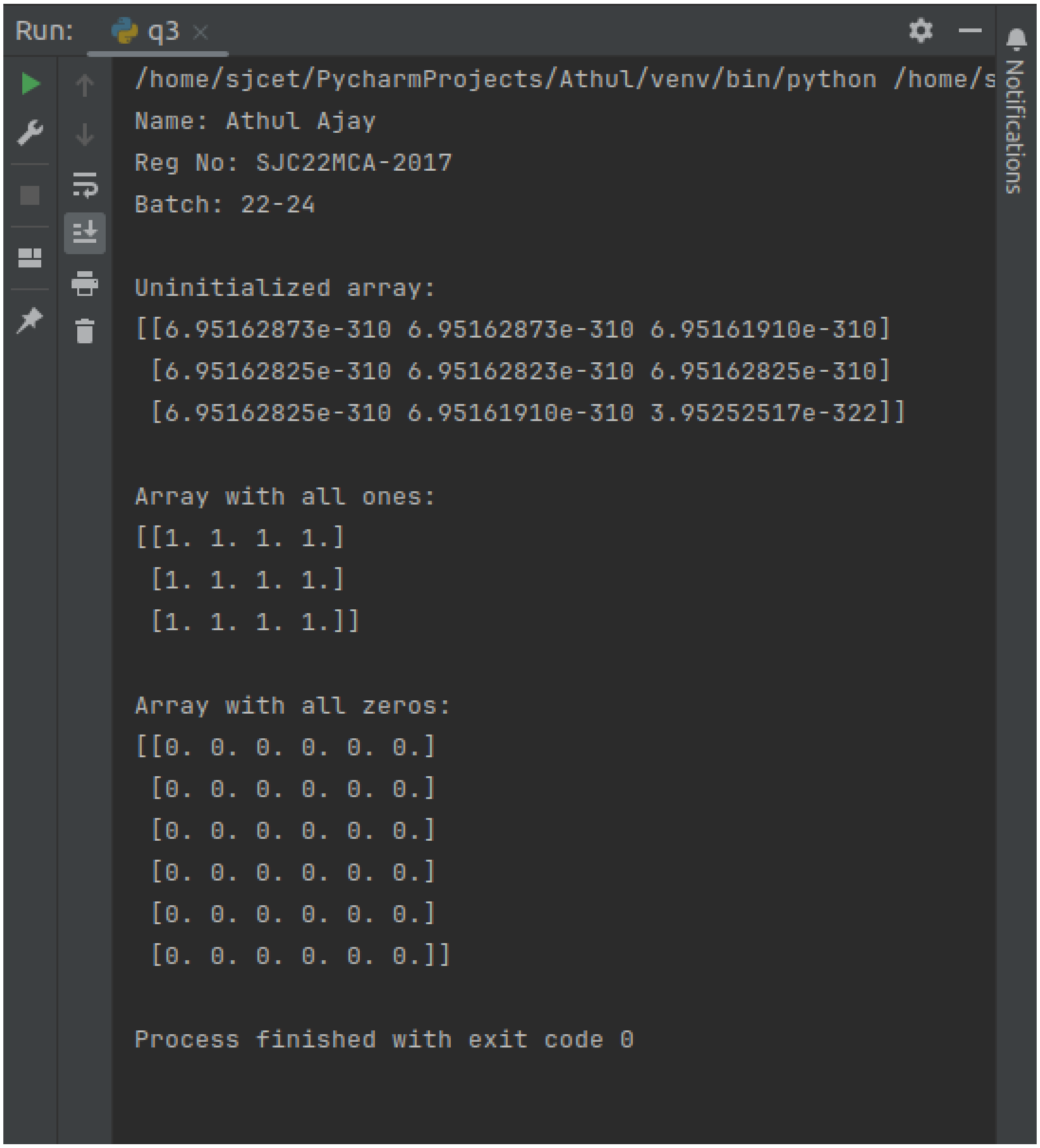
a) an uninitialized array

b) array with all elements as 1,

c) all elements as 0

```
import numpy as np


uninitialized_array = np.empty((3, 3))

ones_array = np.ones((3, 4))

zeros_array = np.zeros((6, 6))
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
print("Uninitialized array:")
print(uninitialized_array)

print("\nArray with all ones:")
print(ones_array)

print("\nArray with all zeros:")
print(zeros_array)
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/s
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Uninitialized array:
[[6.95162873e-310 6.95162873e-310 6.95161910e-310]
 [6.95162825e-310 6.95162823e-310 6.95162825e-310]
 [6.95162825e-310 6.95161910e-310 3.95252517e-322]]

Array with all ones:
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]

Array with all zeros:
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]

Process finished with exit code 0
```

4. Create a one-dimensional array using arange function containing 10 elements.

Display

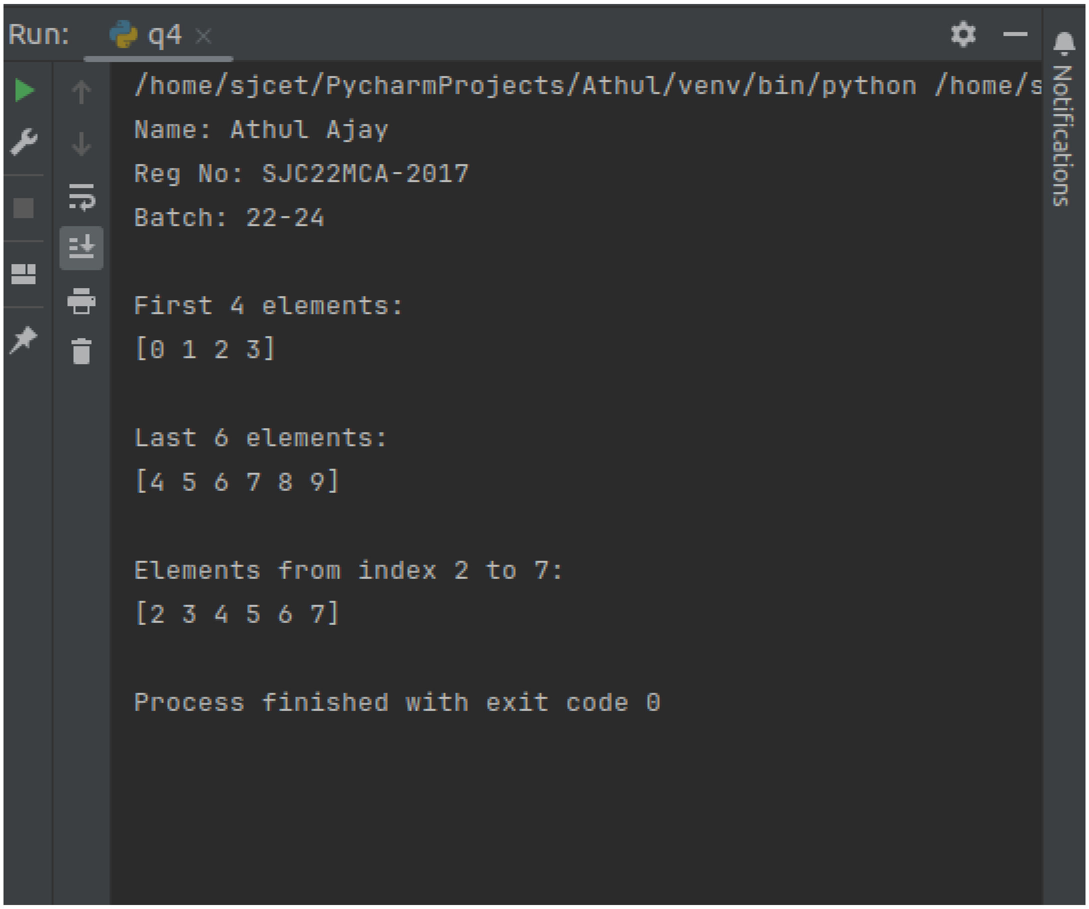
a. First 4 elements

b. Last 6 elements

c. Elements from index 2 to 7

```
import numpy as np

my_array = np.arange(10)
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
print("First 4 elements:")
print(my_array[:4])

print("\nLast 6 elements:")
print(my_array[-6:])

print("\nElements from index 2 to 7:")
print(my_array[2:8]
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/s
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

First 4 elements:
[0 1 2 3]

Last 6 elements:
[4 5 6 7 8 9]

Elements from index 2 to 7:
[2 3 4 5 6 7]

Process finished with exit code 0
```

5. Create an 1D array with arange containing first 15 even numbers as elements

a. Elements from index 2 to 8 with step 2(also demonstrate the same using slice function)

b. Last 3 elements of the array using negative index

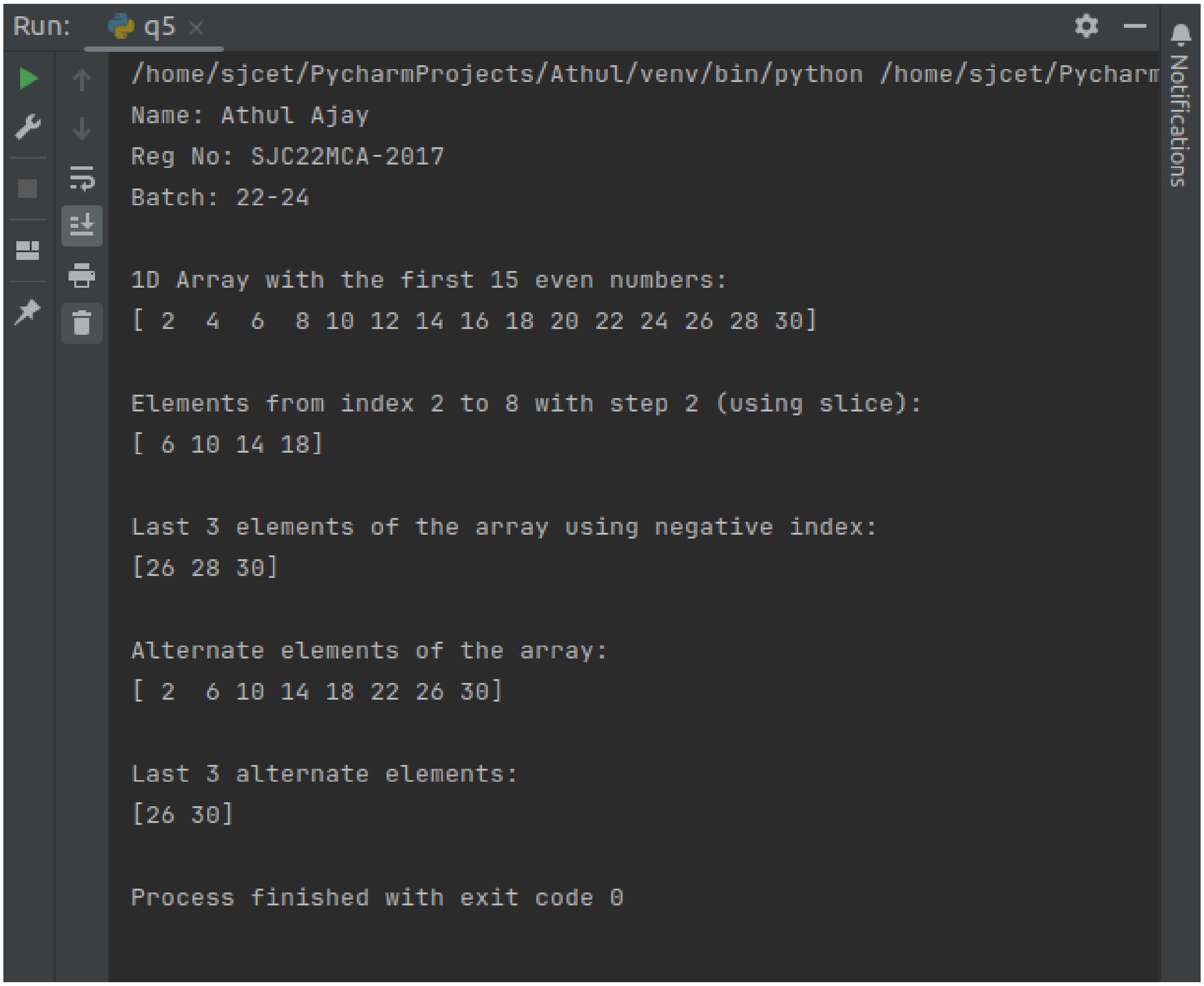
c. Alternate elements of the array

d. Display the last 3 alternate elements

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
even_numbers = np.arange(2, 32, 2)
print("1D Array with the first 15 even numbers:")
print(even_numbers)

subset_a = even_numbers[2:9:2]
print("\nElements from index 2 to 8 with step 2 (using slice):")
print(subset_a)

subset_b = even_numbers[-3:]
print("\nLast 3 elements of the array using negative index:")
print(subset_b)

subset_c = even_numbers[::2]
print("\nAlternate elements of the array:")
print(subset_c)
```

```
subset_d = even_numbers[-3::2]
print("\nLast 3 alternate elements:")
print(subset_d)
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

1D Array with the first 15 even numbers:
[ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30]

Elements from index 2 to 8 with step 2 (using slice):
[ 6 10 14 18]

Last 3 elements of the array using negative index:
[26 28 30]

Alternate elements of the array:
[ 2  6 10 14 18 22 26 30]

Last 3 alternate elements:
[26 30]

Process finished with exit code 0
```

6. Create a 2-Dimensional array with 4 rows and 4 columns.

a. Display all elements excluding the first row

b. Display all elements excluding the last column

c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row

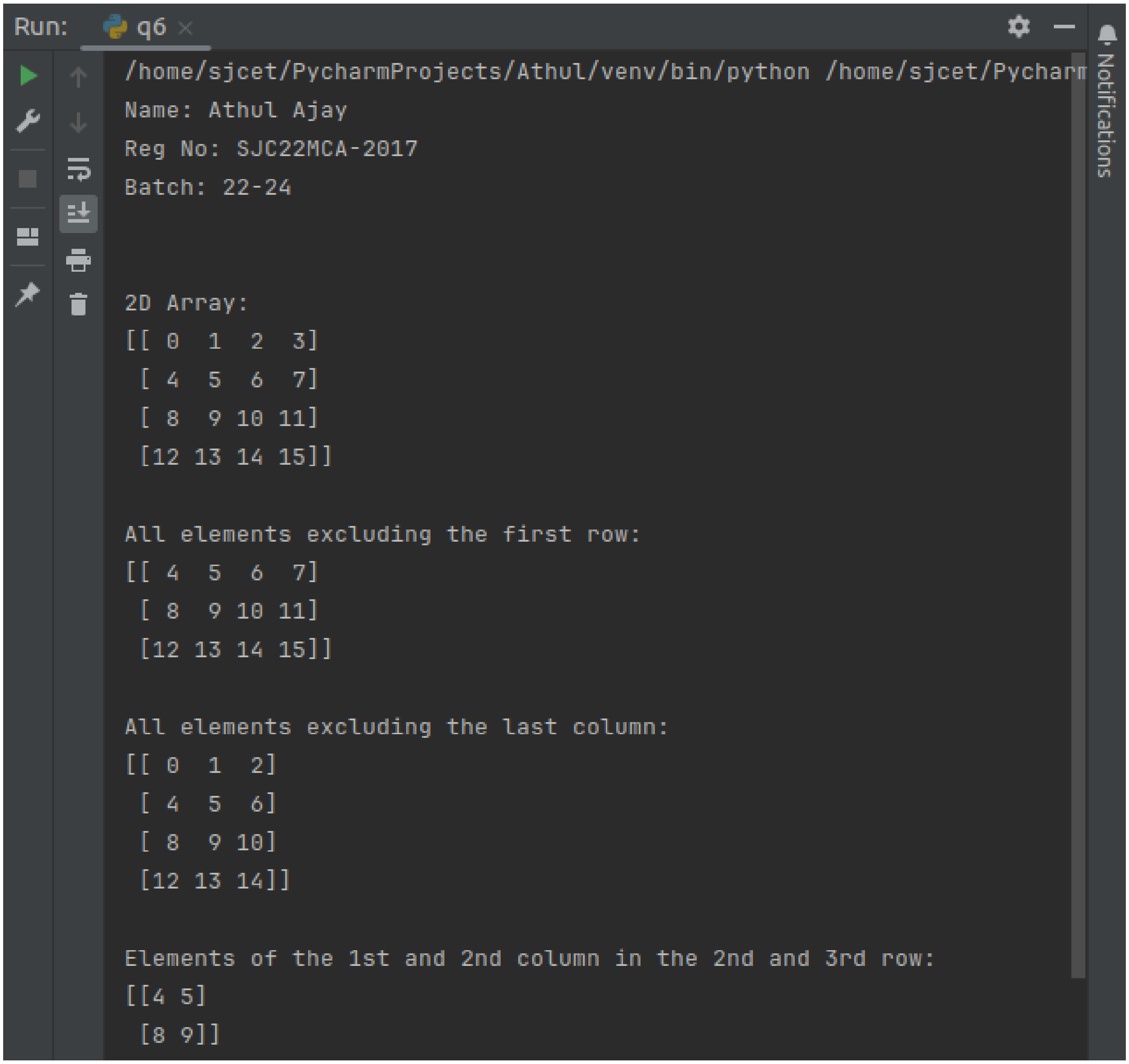
d. Display the elements of 2 nd and 3 rd column

e. Display 2 nd and 3 rd element of 1 st row

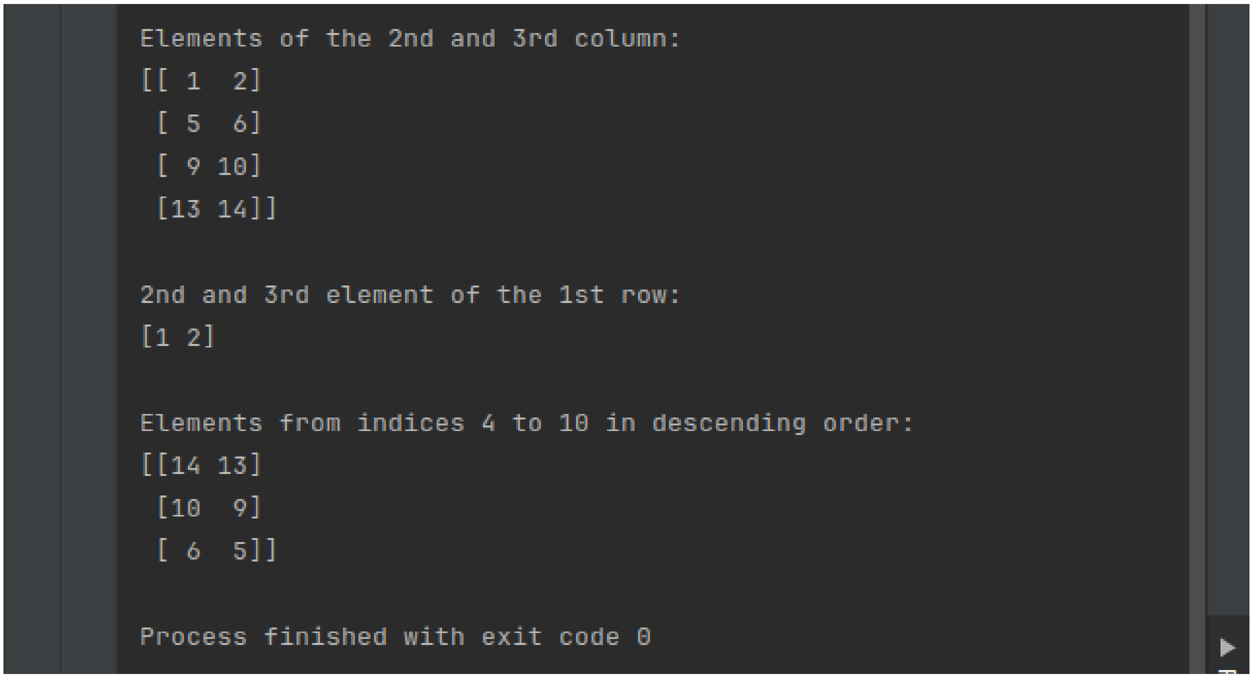
f. Display the elements from indices 4 to 10 in descending order (use–values)

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
two_dim_array = np.arange(16).reshape(4, 4)
print("\n2D Array:")
print(two_dim_array)

print("\nAll elements excluding the first row:")
print(two_dim_array[1:])

print("\nAll elements excluding the last column:")
print(two_dim_array[:, :-1])

print("\nElements of the 1st and 2nd column in the 2nd and 3rd row:")
print(two_dim_array[1:3, 0:2])

print("\nElements of the 2nd and 3rd column:")
print(two_dim_array[:, 1:3])

print("\n2nd and 3rd element of the 1st row:")
print(two_dim_array[0, 1:3])

print("\nElements from indices 4 to 10 in descending order:")
print(two_dim_array[3:0:-1, 2:0:-1])
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24



2D Array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

All elements excluding the first row:
[[ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

All elements excluding the last column:
[[ 0  1  2]
 [ 4  5  6]
 [ 8  9 10]
 [12 13 14]]

Elements of the 1st and 2nd column in the 2nd and 3rd row:
[[4 5]
 [8 9]]
```



```
Elements of the 2nd and 3rd column:
[[ 1  2]
 [ 5  6]
 [ 9 10]
 [13 14]]

2nd and 3rd element of the 1st row:
[1 2]

Elements from indices 4 to 10 in descending order:
[[14 13]
 [10  9]
 [ 6  5]]

Process finished with exit code 0
```

7. Create two 2D arrays using array object and

a. Add the 2 matrices and print it

b. Subtract 2 matrices

c. Multiply the individual elements of matrix

d. Divide the elements of the matrices

e. Perform matrix multiplication

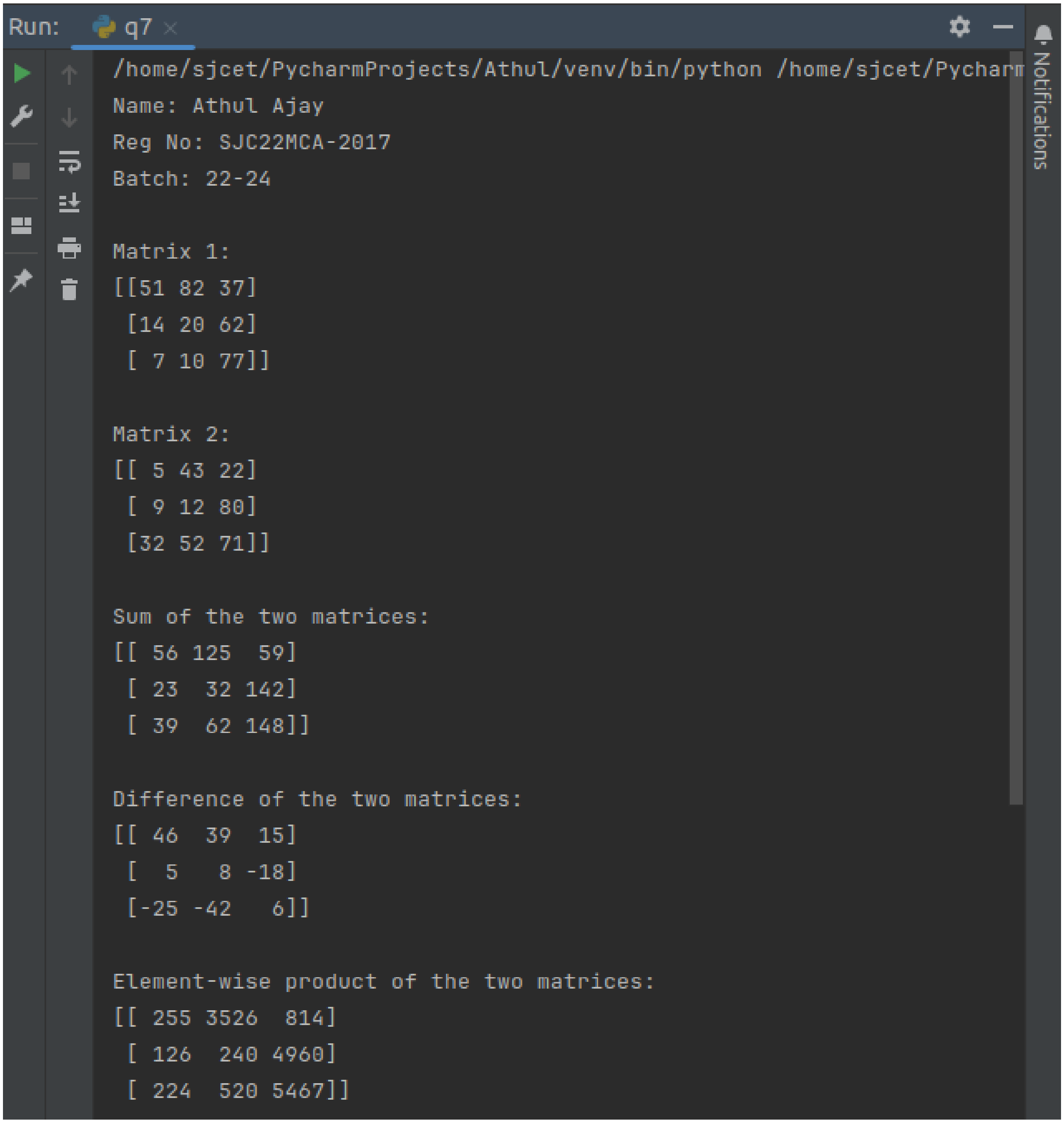
f. Display transpose of the matrix

g. Sum of diagonal elements of a matrix

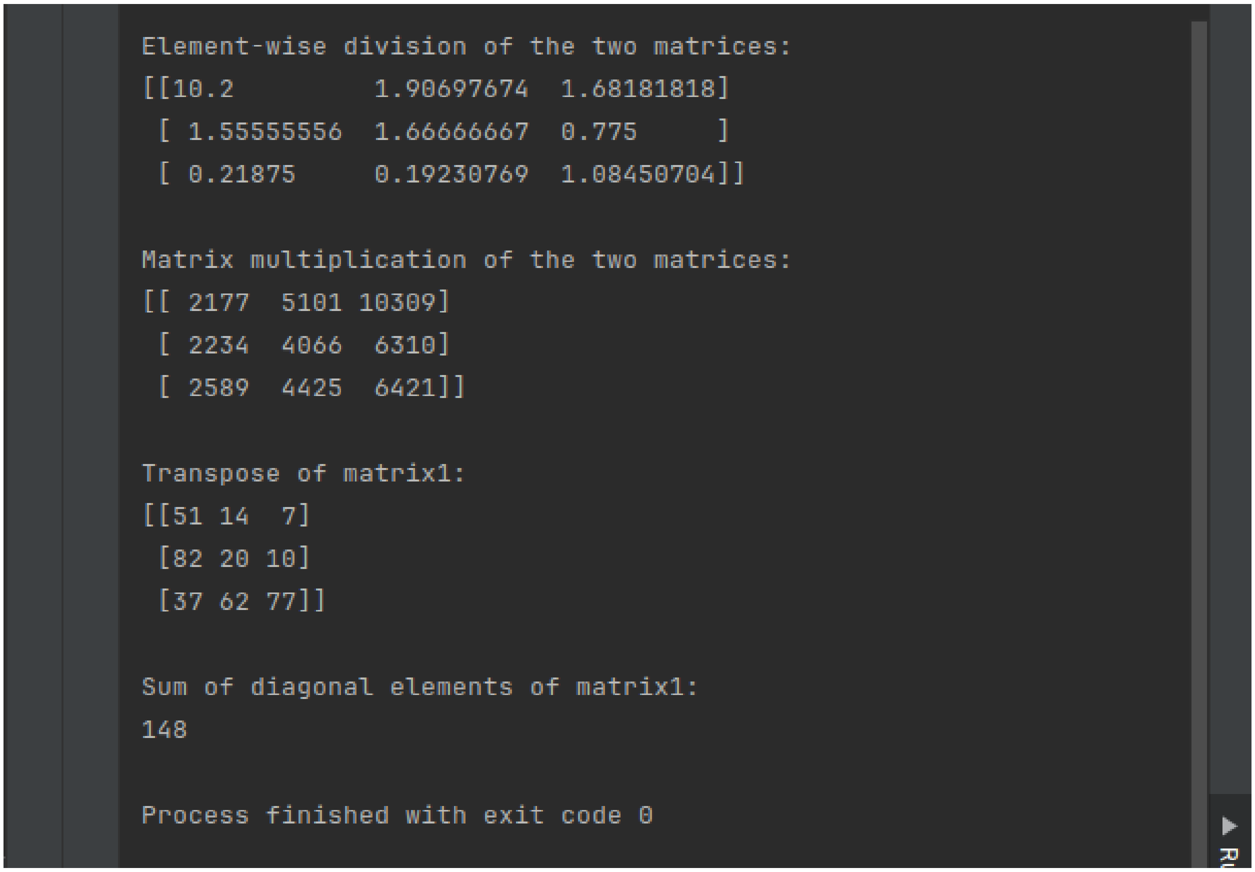
```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
matrix1 = np.array([[51, 82, 37], [14, 20, 62], [7, 10, 77]])
matrix2 = np.array([[5, 43, 22], [9, 12, 80], [32, 52, 71]])
print("Matrix 1:")
print(matrix1)
print()
print("Matrix 2:")
print(matrix2)
print()
matrix_sum = matrix1 + matrix2
print("Sum of the two matrices:")
print(matrix_sum)

matrix_diff = matrix1 - matrix2
```

```
print("\nDifference of the two matrices:")
print(matrix_diff)

matrix_product = matrix1 * matrix2
print("\nElement-wise product of the two matrices:")
print(matrix_product)

with np.errstate(divide='ignore', invalid='ignore'):
    matrix_division = np.true_divide(matrix1, matrix2)
    matrix_division[~np.isfinite(matrix_division)] = np.nan
print("\nElement-wise division of the two matrices:")
print(matrix_division)

matrix_mult = np.dot(matrix1, matrix2)
print("\nMatrix multiplication of the two matrices:")
print(matrix_mult)



matrix1_transpose = np.transpose(matrix1)
print("\nTranspose of matrix1:")
print(matrix1_transpose)

diagonal_sum = np.trace(matrix1)
print("\nSum of diagonal elements of matrix1:")
print(diagonal_sum)
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Matrix 1:
[[51 82 37]
 [14 20 62]
 [ 7 10 77]]

Matrix 2:
[[ 5 43 22]
 [ 9 12 80]
 [32 52 71]]

Sum of the two matrices:
[[ 56 125  59]
 [ 23  32 142]
 [ 39  62 148]]

Difference of the two matrices:
[[ 46  39  15]
 [  5   8 -18]
 [-25 -42   6]]

Element-wise product of the two matrices:
[[ 255 3526  814]
 [ 126  240 4960]
 [ 224  520 5467]]
```

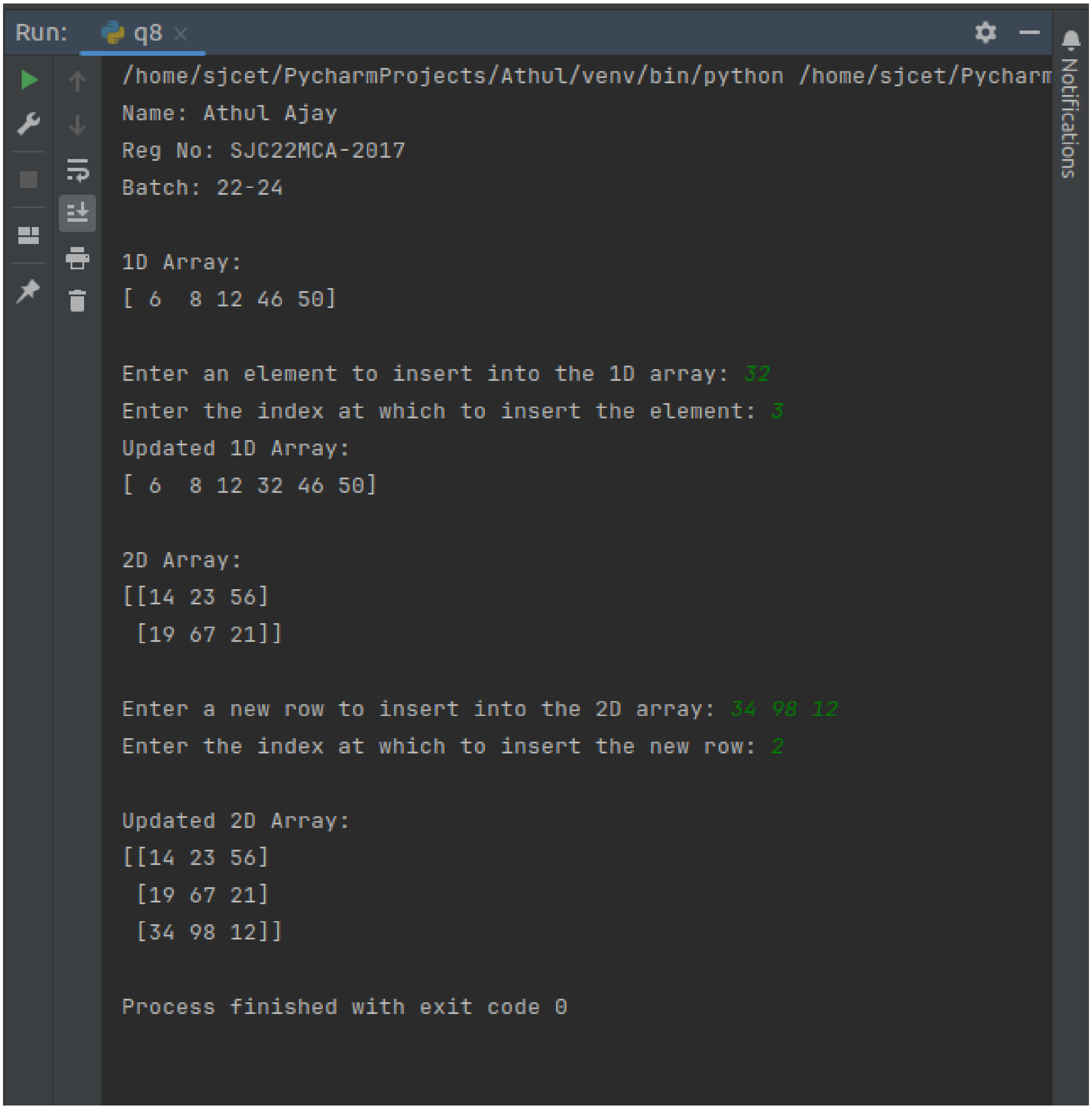


```
Element-wise division of the two matrices:
[[10.2         1.90697674  1.68181818]
 [ 1.55555556  1.66666667  0.775     ]
 [ 0.21875     0.19230769  1.08450704]]

Matrix multiplication of the two matrices:
[[ 2177  5101 10309]
 [ 2234  4066  6310]
 [ 2589  4425  6421]]

Transpose of matrix1:
[[51 14  7]
 [82 20 10]
 [37 62 77]]

Sum of diagonal elements of matrix1:
148

Process finished with exit code 0
```

8. Demonstrate the use of insert () function in 1D and 2D array

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
# 1D Array Example
arr1d = np.array([6, 8, 12, 46, 50])

# Display the 1D array
print("1D Array:")
print(arr1d)
print()
# Take user input for the element and index
element_to_insert_1d = int(input("Enter an element to insert into the 1D array: "))
index_to_insert_1d = int(input("Enter the index at which to insert the element: "))

# Insert the user-specified element at the user-specified index
arr1d = np.insert(arr1d, index_to_insert_1d, element_to_insert_1d)

# Print the updated 1D array
print("Updated 1D Array:")
print(arr1d)
print()
# 2D Array Example
arr2d = np.array([[14, 23, 56], [19, 67, 21]])

# Display the 1D array
print("2D Array:")
print(arr2d)
print()
# Take user input for a new row
new_row = input("Enter a new row to insert into the 2D array: ").split()
new_row = np.array([int(x) for x in new_row])

# Take user input for the index at which to insert the new row
index_to_insert_row = int(input("Enter the index at which to insert the new row: "))

# Insert the user-specified row at the user-specified index along the rows (axis 0)
arr2d = np.insert(arr2d, index_to_insert_row, new_row, axis=0)

# Print the updated 2D array
print("\nUpdated 2D Array:")
print(arr2d)
```

Output:

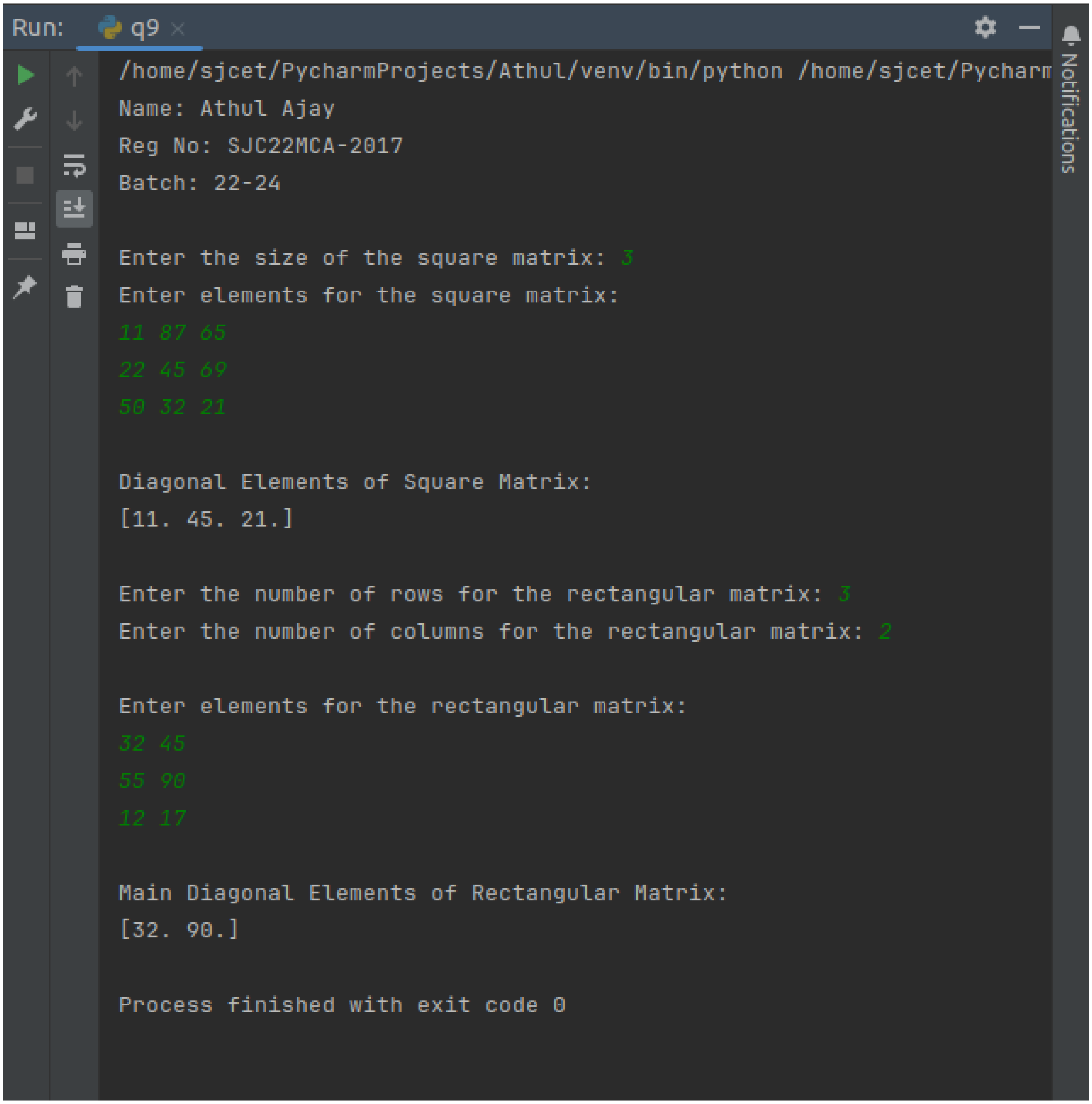


```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

1D Array:
[ 6  8 12 46 50]

Enter an element to insert into the 1D array: 32
Enter the index at which to insert the element: 3
Updated 1D Array:
[ 6  8 12 32 46 50]

2D Array:
[[14 23 56]
 [19 67 21]]

Enter a new row to insert into the 2D array: 34 98 12
Enter the index at which to insert the new row: 2

Updated 2D Array:
[[14 23 56]
 [19 67 21]
 [34 98 12]]

Process finished with exit code 0
```

9. Demonstrate the use of diag () function in 1D and 2D array (use both square matrix and matrix with different dimensions)

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
# Square Matrix Example
n = int(input("Enter the size of the square matrix: "))
square_matrix = np.zeros((n, n))

print("Enter elements for the square matrix:")
for i in range(n):
    row = input().split()
    for j in range(n):
        square_matrix[i][j] = int(row[j])

# Extract the diagonal elements using diag()
diagonal_elements_square = np.diag(square_matrix)

# Print the diagonal elements of the square matrix
print("\nDiagonal Elements of Square Matrix:")
print(diagonal_elements_square)
print()
# Rectangular Matrix Example
m = int(input("Enter the number of rows for the rectangular matrix: "))
n = int(input("Enter the number of columns for the rectangular matrix: "))
rectangular_matrix = np.zeros((m, n))

print("\nEnter elements for the rectangular matrix:")
for i in range(m):
    row = input().split()
    for j in range(n):
        rectangular_matrix[i][j] = int(row[j])

# Extract the main diagonal elements using diag()
main_diagonal_elements_rectangular = np.diag(rectangular_matrix)

# Print the main diagonal elements of the rectangular matrix
print("\nMain Diagonal Elements of Rectangular Matrix:")
print(main_diagonal_elements_rectangular)
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Enter the size of the square matrix: 3
Enter elements for the square matrix:
11 87 65
22 45 69
50 32 21

Diagonal Elements of Square Matrix:
[11. 45. 21.]

Enter the number of rows for the rectangular matrix: 3
Enter the number of columns for the rectangular matrix: 2

Enter elements for the rectangular matrix:
32 45
55 90
12 17

Main Diagonal Elements of Rectangular Matrix:
[32. 90.]

Process finished with exit code 0
```

10. Create a square matrix with random integer values (use randint()) and use

appropriate functions to find:

i) inverse

ii) rank of matrix

iii) Determinant

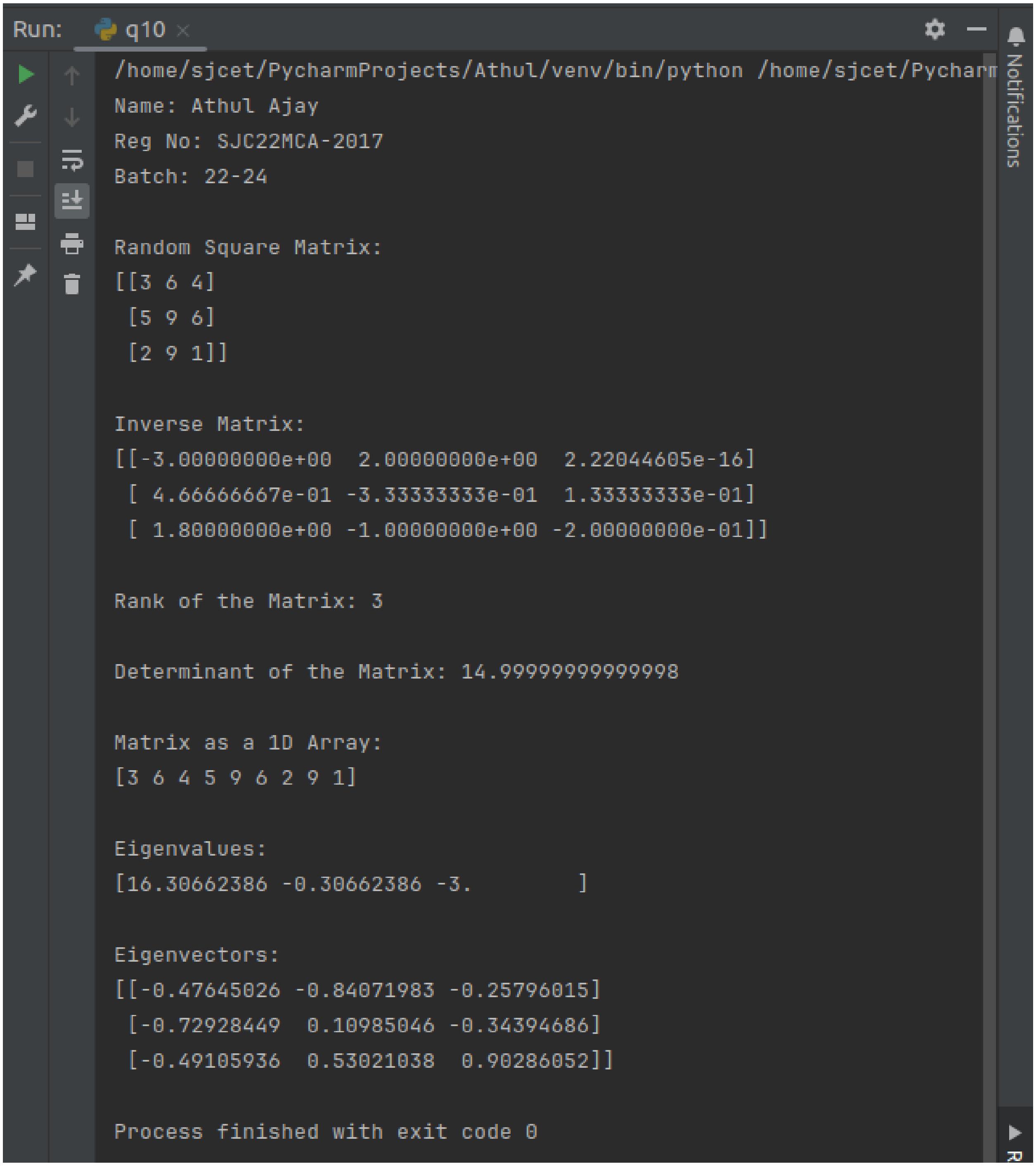
iv) transform matrix into 1D array

v) eigen values and vectors

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
matrix_size = 3

random_matrix = np.random.randint(1, 11, size=(matrix_size, matrix_size))

print("Random Square Matrix:")
print(random_matrix)

try:
    inverse_matrix = np.linalg.inv(random_matrix)
    print("\nInverse Matrix:")
    print(inverse_matrix)
except np.linalg.LinAlgError:
    print("\nInverse does not exist for this matrix.")

rank = np.linalg.matrix_rank(random_matrix)
print("\nRank of the Matrix:", rank)

determinant = np.linalg.det(random_matrix)
print("\nDeterminant of the Matrix:", determinant)

matrix_1d = random_matrix.flatten()
print("\nMatrix as a 1D Array:")
print(matrix_1d)

eigenvalues, eigenvectors = np.linalg.eig(random_matrix)
```

```
print("\nEigenvalues:")
print(eigenvalues)
print("\nEigenvectors:")
print(eigenvectors)
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Random Square Matrix:
[[3 6 4]
 [5 9 6]
 [2 9 1]]

Inverse Matrix:
[[-3.00000000e+00  2.00000000e+00  2.22044605e-16]
 [ 4.66666667e-01 -3.33333333e-01  1.33333333e-01]
 [ 1.80000000e+00 -1.00000000e+00 -2.00000000e-01]]

Rank of the Matrix: 3

Determinant of the Matrix: 14.999999999999998

Matrix as a 1D Array:
[3 6 4 5 9 6 2 9 1]

Eigenvalues:
[16.30662386 -0.30662386 -3.        ]

Eigenvectors:
[[-0.47645026 -0.84071983 -0.25796015]
 [-0.72928449  0.10985046 -0.34394686]
 [-0.49105936  0.53021038  0.90286052]]

Process finished with exit code 0
```

11.a. Create a matrix X with suitable rows and columns

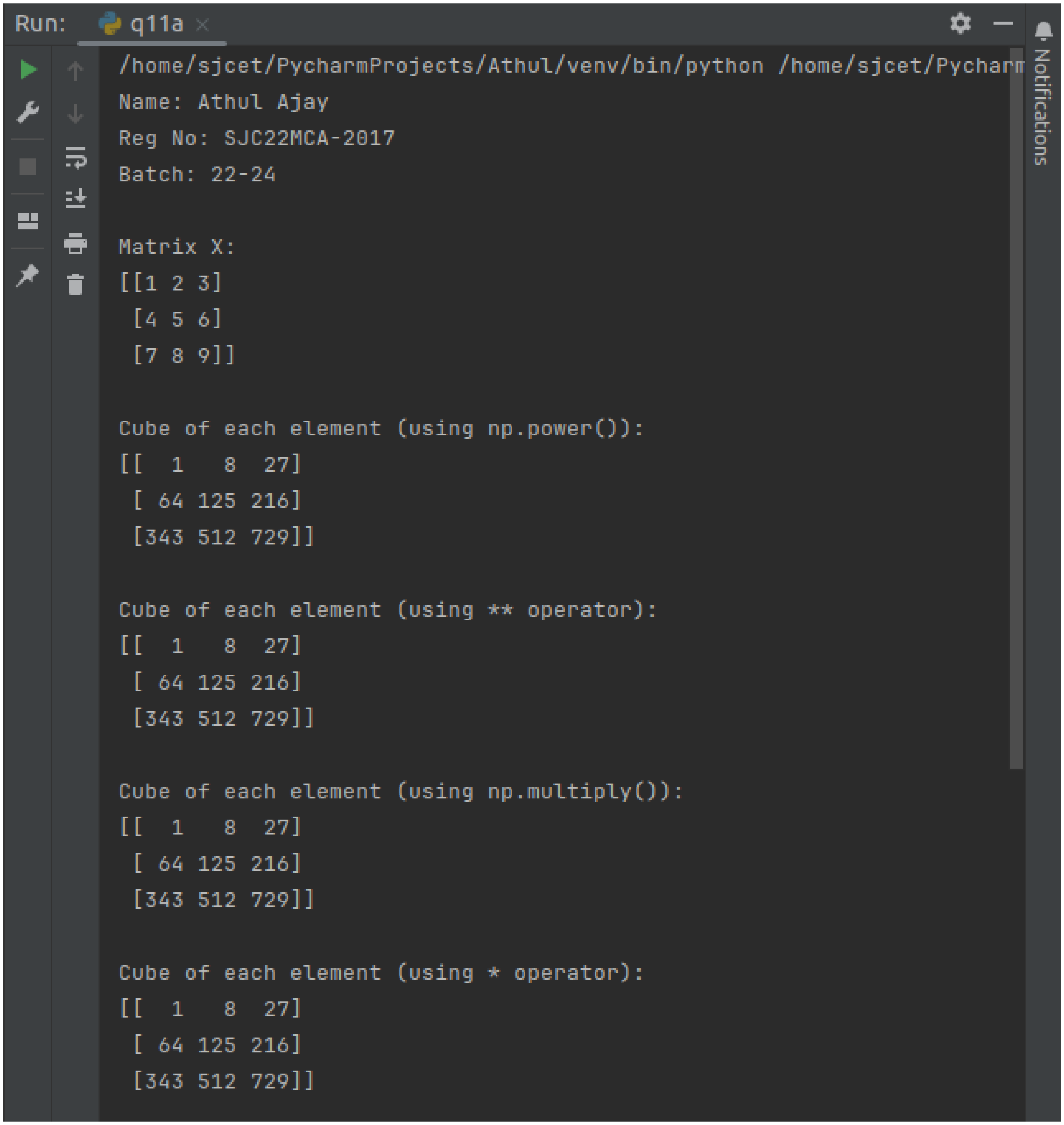
i) Display the cube of each element of the matrix using different methods (use multiply (), *, power (), **)

ii) Display identity matrix of the given square matrix.

iii) Display each element of the matrix to different powers.

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# i) Display the cube of each element of the matrix using different methods

# Using np.power() to calculate the cube
cubed_matrix1 = np.power(X, 3)

# Using the ** operator to calculate the cube
cubed_matrix2 = X ** 3

# Using np.multiply() to calculate the cube
cubed_matrix3 = np.multiply(X, np.multiply(X, X))

# Using the * operator to calculate the cube
cubed_matrix4 = X * X * X

print("Matrix X:")
print(X)

print("\nCube of each element (using np.power()):")
print(cubed_matrix1)

print("\nCube of each element (using ** operator):")
print(cubed_matrix2)

print("\nCube of each element (using np.multiply()):")
print(cubed_matrix3)
print("\nCube of each element (using * operator):")
print(cubed_matrix4)

# ii) Display the identity matrix of the given square matrix
identity_matrix = np.identity(X.shape[0])
print("\nIdentity Matrix of X:")
print(identity_matrix)

# iii) Display each element of the matrix to different powers
exponentials = [2, 3, 4]

powered_matrices = [np.power(X, exp) for exp in exponentials]

for i, exp in enumerate(exponentials):
    print(f"\nMatrix X to the power of {exp}:")
    print(powered_matrices[i])
```

Output:

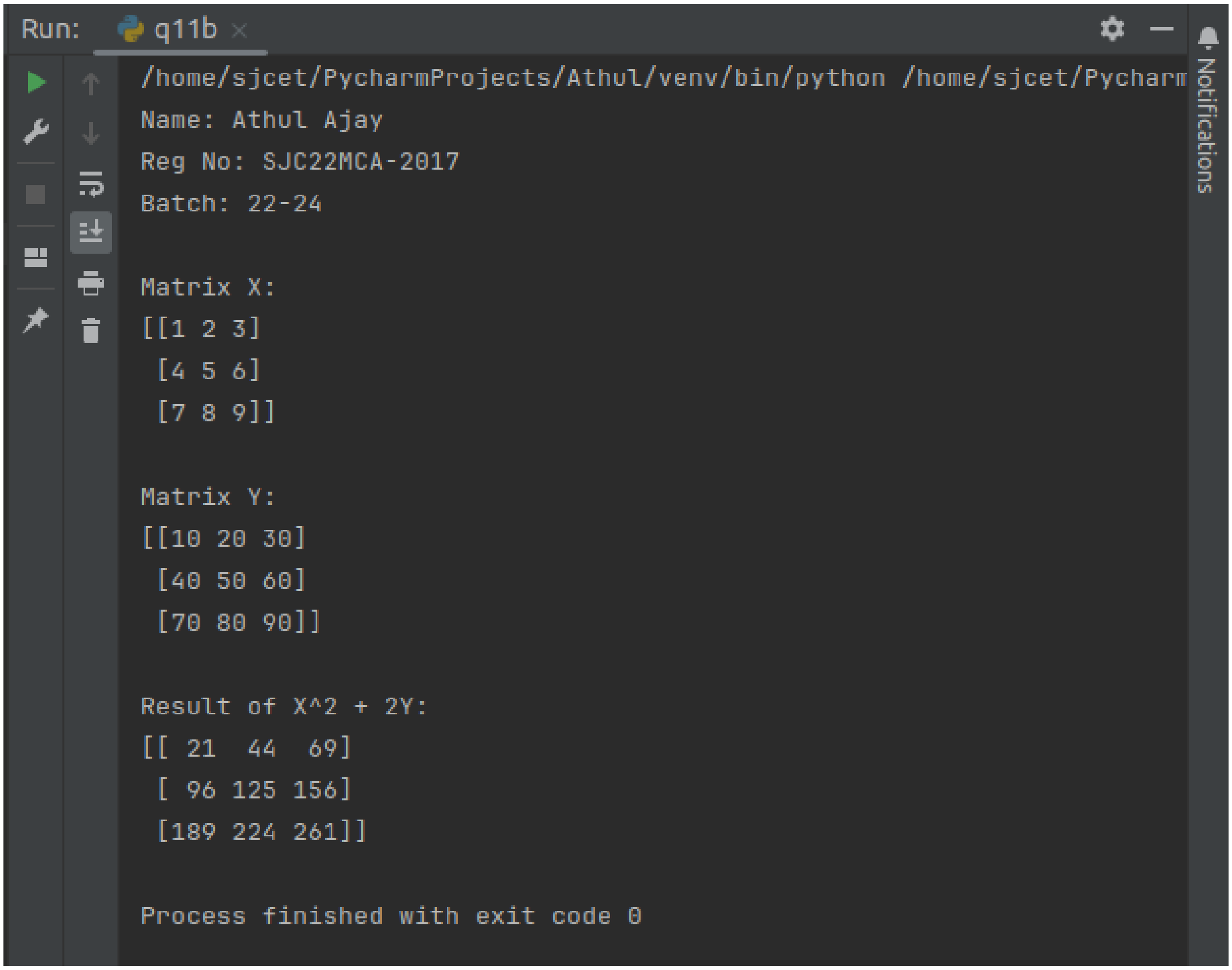


```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Matrix X:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Cube of each element (using np.power()):
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]

Cube of each element (using ** operator):
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]

Cube of each element (using np.multiply()):
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]

Cube of each element (using * operator):
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]
```

11.b. Create a matrix Y with same dimension as X and perform the operation X 2 +2Y

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

Y = np.array([[10, 20, 30],
              [40, 50, 60],
              [70, 80, 90]])

result = np.power(X, 2) + 2 * Y

print("Matrix X:")
print(X)

print("\nMatrix Y:")
print(Y)

print("\nResult of X^2 + 2Y:")
print(result)
```

Output:

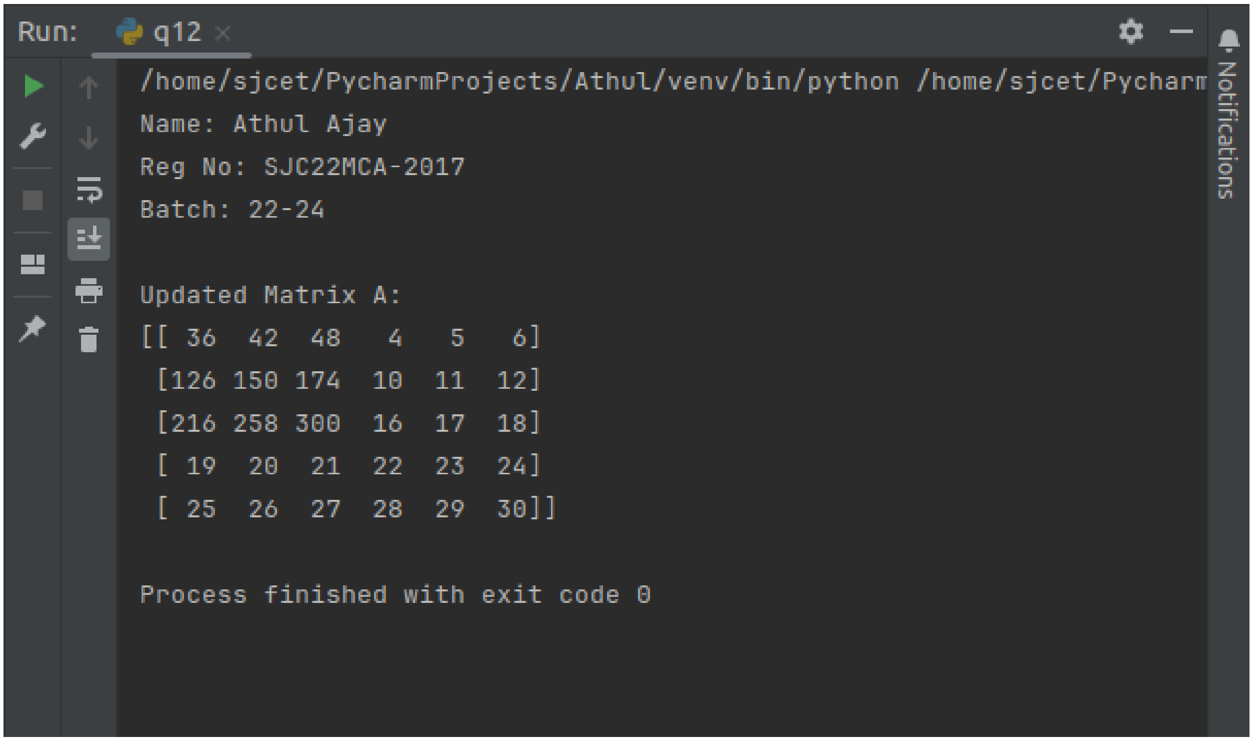


```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Matrix X:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Matrix Y:
[[10 20 30]
 [40 50 60]
 [70 80 90]]

Result of X^2 + 2Y:
[[ 21  44  69]
 [ 96 125 156]
 [189 224 261]]

Process finished with exit code 0
```

12. Define matrices A with dimension 5x6 and B with dimension 3x3. Extract a sub matrix of dimension 3x3 from A and multiply it with B. Replace the extracted sub matrix in A with the matrix obtained after multiplication

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
A = np.array([[1, 2, 3, 4, 5, 6],
              [7, 8, 9, 10, 11, 12],
              [13, 14, 15, 16, 17, 18],
              [19, 20, 21, 22, 23, 24],
              [25, 26, 27, 28, 29, 30]])

B = np.array([[2, 3, 4],
              [5, 6, 7],
              [8, 9, 10]])

submatrix_A = A[:3, :3]

result = np.dot(submatrix_A, B)

A[:3, :3] = result

# Display the updated matrix A
print("Updated Matrix A:")
print(A)
```

Output:

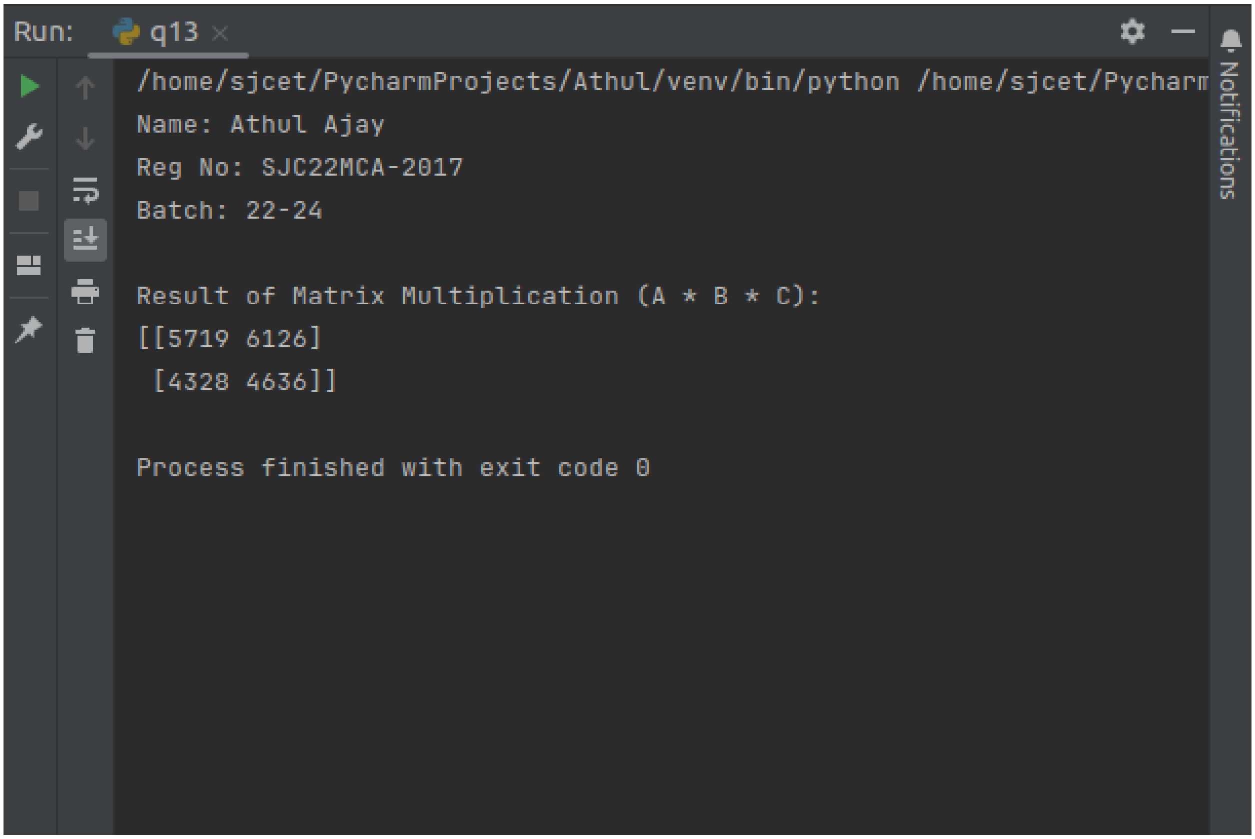


```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Updated Matrix A:
[[ 36  42  48   4   5   6]
 [126 150 174  10  11  12]
 [216 258 300  16  17  18]
 [ 19  20  21  22  23  24]
 [ 25  26  27  28  29  30]]

Process finished with exit code 0
```

13. Given 3 Matrices A, B and C. Write a program to perform matrix multiplication of the 3 matrices.

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
A = np.array([[7, 5, 9],
           [2, 11, 3]])

B = np.array([[7, 8],
           [9, 10],
           [11, 12]])

C = np.array([[13, 14],
           [15, 16]])

result = np.dot(np.dot(A, B), C)

print("Result of Matrix Multiplication (A * B * C):")
print(result)
```

Output:

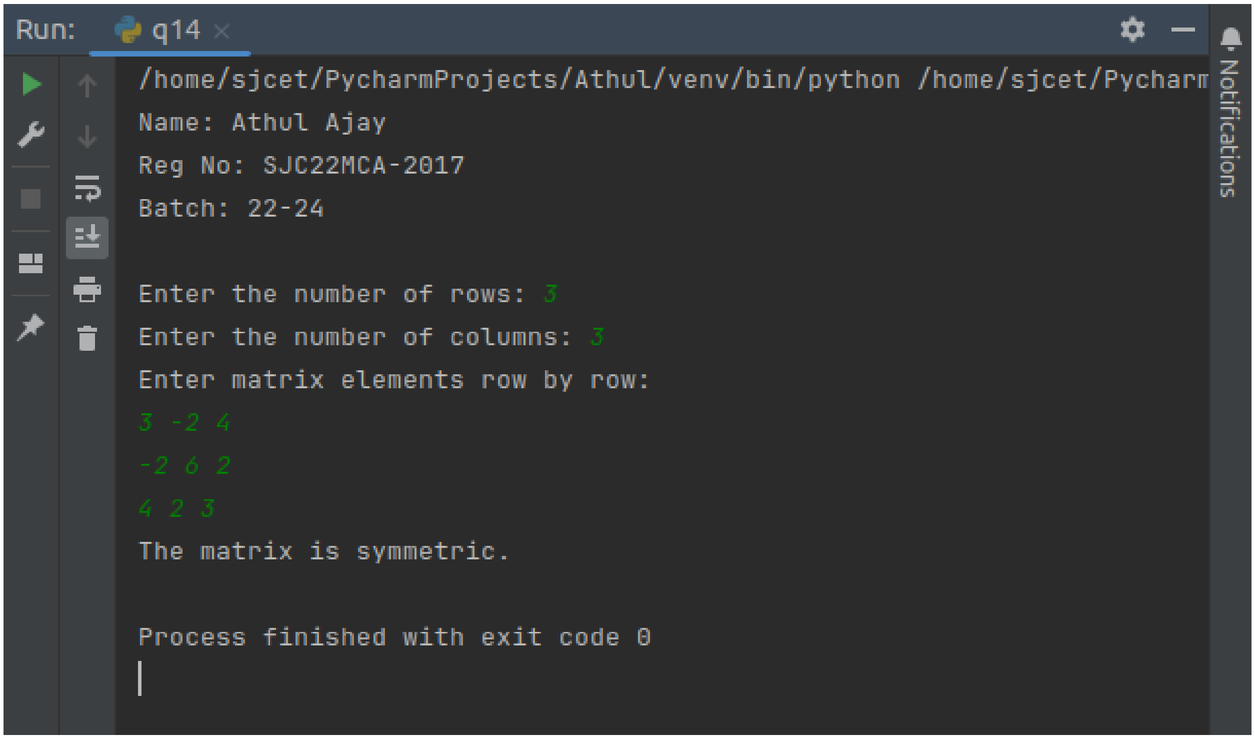


```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Result of Matrix Multiplication (A * B * C):
[[5719 6126]
 [4328 4636]]

Process finished with exit code 0
```

14. Write a program to check whether given matrix is symmetric or Skew Symmetric.

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
# Function to check if a matrix is symmetric
def is_symmetric(matrix):
    return np.array_equal(matrix, matrix.T)

# Function to check if a matrix is skew-symmetric
def is_skew_symmetric(matrix):
    return np.array_equal(matrix, -matrix.T)

# Input matrix dimensions
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))

# Initialize an empty matrix
matrix = np.zeros((rows, cols))

print("Enter matrix elements row by row:")
for i in range(rows):
    row = input().split()
    for j in range(cols):
        matrix[i][j] = float(row[j])

# Check if the matrix is symmetric or skew-symmetric
if is_symmetric(matrix):
    print("The matrix is symmetric.")
elif is_skew_symmetric(matrix):
    print("The matrix is skew-symmetric.")
else:
    print("The matrix is neither symmetric nor skew-symmetric."
```

Output:



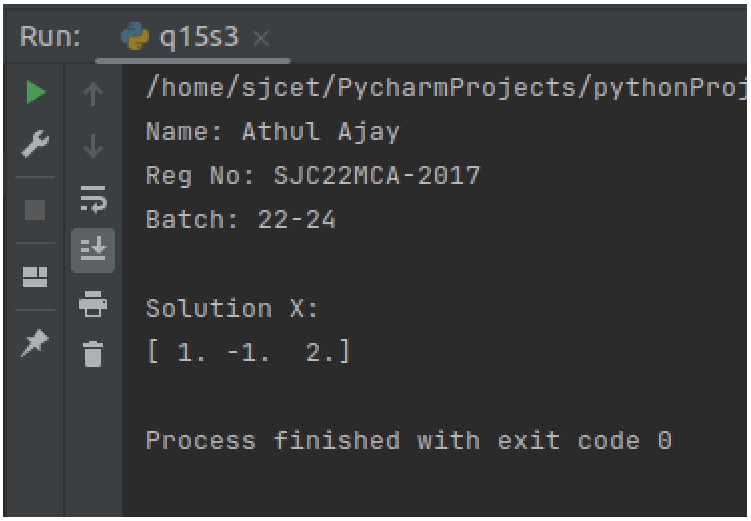
```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Enter the number of rows: 3
Enter the number of columns: 3
Enter matrix elements row by row:
3 -2 4
-2 6 2
4 2 3
The matrix is symmetric.

Process finished with exit code 0
```

15. Given a matrix-vector equation AX=b. Write a program to find out the value of X using solve

Note: Numpy provides a function called solve for solving such equations.

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
A = np.array([[2, 1, -2],
              [3, 0, 1],
              [1, 1, -1]])

b = np.array([-3, 5, 2])

try:

    X = np.linalg.solve(A, b)


    print("Solution X:")
    print(X)
except np.linalg.LinAlgError:
    print("Matrix A is singular. The system of equations may not have a unique solution.")
```

Output:

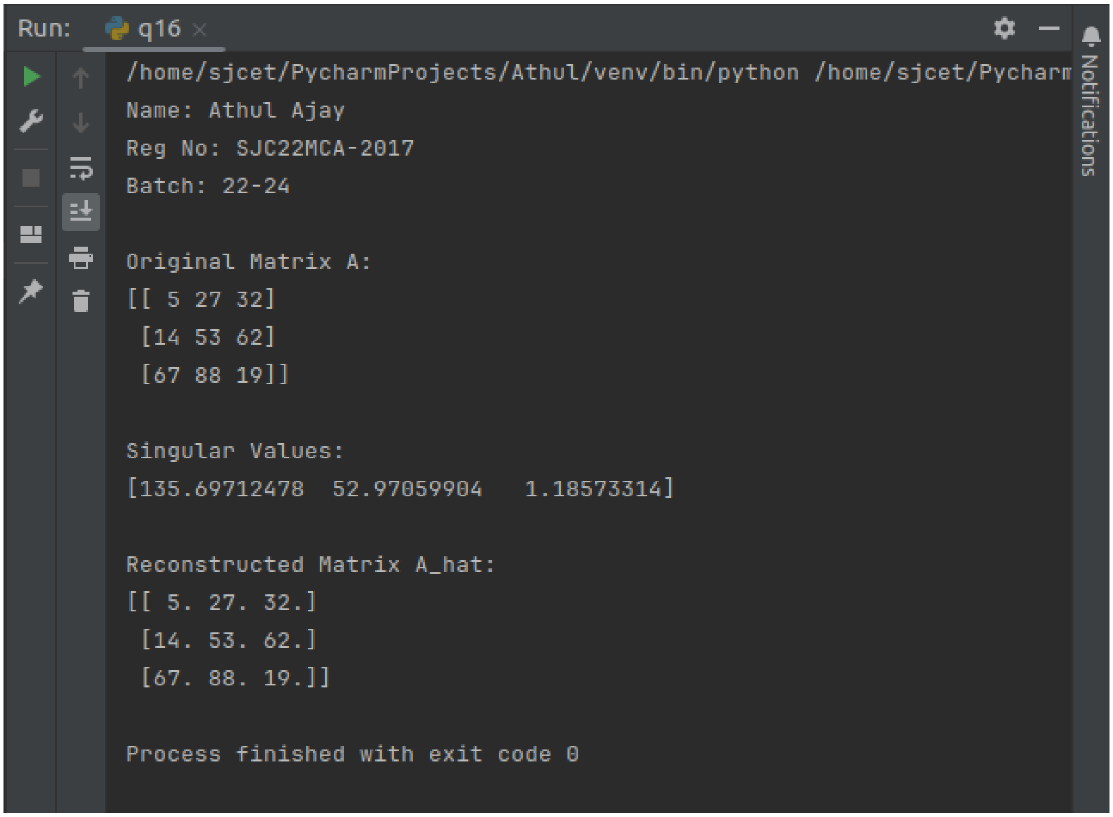


```
/home/sjcet/PycharmProjects/pythonProj
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Solution X:
[ 1. -1.  2.]

Process finished with exit code 0
```

16. Write a program to perform the SVD of a given matrix A. Also reconstruct the given matrix from the 3 matrices obtained after performing SVD. Use the function: numpy.linalg.svd ()

```
import numpy as np
print("Name: Athul Ajay")
print("Reg No: SJC22MCA-2017")
print("Batch: 22-24")
print()
A = np.array([[5, 27, 32], [14, 53, 62], [67, 88, 19]])

U, S, Vt = np.linalg.svd(A)

A_hat = U @ np.diag(S) @ Vt

print("Original Matrix A:")
print(A)
print("\nSingular Values:")
print(S)
print("\nReconstructed Matrix A_hat:")
print(A_hat)
```

Output:



```
/home/sjcet/PycharmProjects/Athul/venv/bin/python /home/sjcet/Pycharm
Name: Athul Ajay
Reg No: SJC22MCA-2017
Batch: 22-24

Original Matrix A:
[[ 5 27 32]
 [14 53 62]
 [67 88 19]]

Singular Values:
[135.69712478  52.97059904   1.18573314]

Reconstructed Matrix A_hat:
[[ 5. 27. 32.]
 [14. 53. 62.]
 [67. 88. 19.]]

Process finished with exit code 0
```